INTEGRATION

Obtention de l'API

Se rendre sur le site officiel de VirusTotal, se créer un compte, l'API apparaît dans les settings :

API Key

Limited , standard free public API *Upgrade to premium*	
Usage	Must not be used in business workflows, commercial products or services.
Request rate	4 lookups / min
Daily quota	500 lookups / day
Monthly quota	15.5 K lookups / month

Intégration dans Wazuh

Remarque : Effectuer cette commande les manipulation > /var/ossec/logs/ossec.log , cela permettra de vérifier, plus facilement, au redémarrage après intégration les logs du serveur.

Depuis Wazuh Server :

nano /var/ossec/etc/ossec.conf

```
<integration>
  <name>virustotal</name>
  <api_key><VIRUSTOTAL_API_KEY></api_key> <!-- Replace with your VirusTotal API key
-->
  <group>syscheck</group>
  <alert_format>json</alert_format>
</integration>
```

Utilité

L'ajout de la balise "integration" permet à Wazuh d'établir la connexion entre VirusTotal et le SIEM.

Lorsque Wazuh détectera, grâce à son module FIM (File Integrity Monitoring) une modification, une alterte sera générée contenant les informations du fichier, **y compris son hash**.

En indiquant le group "syscheck", on demande au serveur de déclencher l'intégration après chaques alertes générées par un module de surveillance de fichier. Dans notre cas, FIM.

Un script est déjà intégré par défaut dans /var/ossec/integrations/virustotal.py

A chaque alertes générées par le FIM, Wazuh lis le dossier "integrations" et recherche le name "virustotal" pour l'exécuter.

```
root@wazu02:/var/ossec/integrations# ls
maltiverse  maltiverse.py  pagerduty  pagerduty.py  shuffle  shuffle.py  slack
slack.py  virustotal  virustotal.py
```

Analyse script

On peut voir deux script virustotal, officiellement, la balise integration appelle uniquement le script "virustotal". Si on cat ce script :

```
root@DEBwazu02:/var/ossec/integrations# cat virustotal
#!/bin/sh
# Copyright (C) 2015, Wazuh Inc.
# Created by Wazuh, Inc. <info@wazuh.com>.
# This program is free software; you can redistribute it and/or modify it under the
terms of GPLv2

WPYTHON_BIN="framework/python/bin/python3"

SCRIPT_PATH_NAME="$0"

DIR_NAME="$(cd $(dirname ${SCRIPT_PATH_NAME}); pwd -P)"
SCRIPT_NAME="$(basename ${SCRIPT_PATH_NAME})"

case ${DIR_NAME} in
    */active-response/bin | */wodles*)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/../..; pwd)"
        fi

        PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
    ;;
    */bin)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
        fi

        PYTHON_SCRIPT="${WAZUH_PATH}/framework/scripts/$(echo ${SCRIPT_NAME} | sed
's/\-/_/g').py"
    ;;
     */integrations)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
        fi

        PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
    ;;
esac


${WAZUH_PATH}/${WPYTHON_BIN} ${PYTHON_SCRIPT} "$@"
```

Actions réalisées par le script shell :

1. Détection du chemin de l'exécutable Python :
    - Le script identifie l'interpréteur Python utilisé par Wazuh, défini comme `framework/python/bin/python3`.

2. Détermination du chemin du script Python :
    - En fonction de l'emplacement du wrapper shell (`active-response/bin` , `wodles` , `bin` , ou `integrations`), il construit dynamiquement le chemin vers le fichier Python associé.
3. Exécution du script Python :
    - Une fois le chemin déterminé, le wrapper exécute le script Python avec l'interpréteur Python de Wazuh, tout en passant les arguments reçus.

cat du virustotal.py :

```
root@DEBwazu02:/var/ossec/integrations# cat virustotal.py
# Copyright (C) 2015, Wazuh Inc.
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.


import json
import os
import re
import sys
from socket import AF_UNIX, SOCK_DGRAM, socket

# Exit error codes
ERR_NO_REQUEST_MODULE = 1
ERR_BAD_ARGUMENTS = 2
ERR_BAD_MD5_SUM = 3
ERR_NO_RESPONSE_VT = 4
ERR_SOCKET_OPERATION = 5
ERR_FILE_NOT_FOUND = 6
ERR_INVALID_JSON = 7

try:
    import requests
    from requests.exceptions import Timeout
except Exception:
    print("No module 'requests' found. Install: pip install requests")
    sys.exit(ERR_NO_REQUEST_MODULE)

# ossec.conf configuration:
# <integration>
#   <name>virustotal</name>
#   <api_key>API_KEY</api_key> <!-- Replace with your VirusTotal API key -->
#   <group>syscheck</group>
#   <alert_format>json</alert_format>
# </integration>

# Global vars
debug_enabled = False
timeout = 10
retries = 3
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}

# Log and socket path
LOG_FILE = f'{pwd}/logs/integrations.log'
SOCKET_ADDR = f'{pwd}/queue/sockets/queue'

# Constants
ALERT_INDEX = 1
```

```
APIKEY_INDEX = 2
TIMEOUT_INDEX = 6
RETRIES_INDEX = 7


def main(args):
    global debug_enabled
    global timeout
    global retries
    try:
        # Read arguments
        bad_arguments: bool = False
        msg = ''
        if len(args) >= 4:
            debug_enabled = len(args) > 4 and args[4] == 'debug'
            if len(args) > TIMEOUT_INDEX:
                timeout = int(args[TIMEOUT_INDEX])
            if len(args) > RETRIES_INDEX:
                retries = int(args[RETRIES_INDEX])
        else:
            msg = '# Error: Wrong arguments\n'
            bad_arguments = True

        # Logging the call
        with open(LOG_FILE, 'a') as f:
            f.write(msg)

        if bad_arguments:
            debug('# Error: Exiting, bad arguments. Inputted: %s' % args)
            sys.exit(ERR_BAD_ARGUMENTS)

        # Core function
        process_args(args)

    except Exception as e:
        debug(str(e))
        raise


def process_args(args) -> None:
    """This is the core function, creates a message with all valid fields
    and overwrite or add with the optional fields

    Parameters
    ----------
    args : list[str]
        The argument list from main call
    """
    debug('# Running VirusTotal script')

    # Read args
    alert_file_location: str = args[ALERT_INDEX]
    apikey: str = args[APIKEY_INDEX]

    # Load alert. Parse JSON object.
    json_alert = get_json_alert(alert_file_location)
    debug(f"# Opening alert file at '{alert_file_location}' with '{json_alert}'")

    # Request VirusTotal info
    debug('# Requesting VirusTotal information')
    msg: any = request_virustotal_info(json_alert, apikey)

    if not msg:
        debug('# Error: Empty message')
        raise Exception
```

```
    send_msg(msg, json_alert['agent'])


def debug(msg: str) -> None:
    """Log the message in the log file with the timestamp, if debug flag
    is enabled

    Parameters
    ----------
    msg : str
        The message to be logged.
    """
    if debug_enabled:
        print(msg)
        with open(LOG_FILE, 'a') as f:
            f.write(msg + '\n')


def request_info_from_api(alert, alert_output, api_key):
    """Request information from an API using the provided alert and API key.

    Parameters
    ----------
    alert : dict
        The alert dictionary containing information for the API request.
    alert_output : dict
        The output dictionary where API response information will be stored.
    api_key : str
        The API key required for making the API request.

    Returns
    -------
    dict
        The response data received from the API.

    Raises
    ------
    Timeout
        If the API request times out.
    Exception
        If an unexpected exception occurs during the API request.
    """
    for attempt in range(retries + 1):
        try:
            vt_response_data = query_api(alert['syscheck']['md5_after'], api_key)
            return vt_response_data
        except Timeout:
            debug('# Error: Request timed out. Remaining retries: %s' % (retries -
attempt))
            continue
        except Exception as e:
            debug(str(e))
            sys.exit(ERR_NO_RESPONSE_VT)

    debug('# Error: Request timed out and maximum number of retries was exceeded')
    alert_output['virustotal']['error'] = 408
    alert_output['virustotal']['description'] = 'Error: API request timed out'
    send_msg(alert_output)
    sys.exit(ERR_NO_RESPONSE_VT)


def request_virustotal_info(alert: any, apikey: str):
    """Generate the JSON object with the message to be send
```

```
    Parameters
    ----------
    alert : any
        JSON alert object.
    apikey : str
        The API key required for making the API request.

    Returns
    -------
    msg: str
        The JSON message to send
    """
    alert_output = {'virustotal': {}, 'integration': 'virustotal'}

    # If there is no syscheck block present in the alert. Exit.
    if 'syscheck' not in alert:
        debug('# No syscheck block present in the alert')
        return None

    # If there is no md5 checksum present in the alert. Exit.
    if 'md5_after' not in alert['syscheck']:
        debug('# No md5 checksum present in the alert')
        return None

    # If the md5_after field is not a md5 hash checksum. Exit
    if not (
        isinstance(alert['syscheck']['md5_after'], str) is True
        and len(re.findall(r'\b([a-f\d]{32}|[A-F\d]{32})\b', alert['syscheck']
['md5_after'])) == 1
    ):
        debug('# md5_after field in the alert is not a md5 hash checksum')
        return None

    # Request info using VirusTotal API
    vt_response_data = request_info_from_api(alert, alert_output, apikey)

    alert_output['virustotal']['found'] = 0
    alert_output['virustotal']['malicious'] = 0
    alert_output['virustotal']['source'] = {
        'alert_id': alert['id'],
        'file': alert['syscheck']['path'],
        'md5': alert['syscheck']['md5_after'],
        'sha1': alert['syscheck']['sha1_after'],
    }

    # Check if VirusTotal has any info about the hash
    if in_database(vt_response_data, hash):
        alert_output['virustotal']['found'] = 1

    # Info about the file found in VirusTotal
    if alert_output['virustotal']['found'] == 1:
        if vt_response_data['positives'] > 0:
            alert_output['virustotal']['malicious'] = 1

        # Populate JSON Output object with VirusTotal request
        alert_output['virustotal'].update(
            {
                'sha1': vt_response_data['sha1'],
                'scan_date': vt_response_data['scan_date'],
                'positives': vt_response_data['positives'],
                'total': vt_response_data['total'],
                'permalink': vt_response_data['permalink'],
            }
        )
```

```
    return alert_output


def in_database(data, hash):
    result = data['response_code']
    if result == 0:
        return False
    return True


def query_api(hash: str, apikey: str) -> any:
    """Send a request to VT API and fetch information to build message

    Parameters
    ----------
    hash : str
        Hash need it for parameters
    apikey: str
        Authentication API

    Returns
    -------
    data: any
        JSON with the response

    Raises
    ------
    Exception
        If the status code is different than 200.
    """
    params = {'apikey': apikey, 'resource': hash}
    headers = {'Accept-Encoding': 'gzip, deflate', 'User-Agent': 'gzip,  Python
library-client-VirusTotal'}

    debug('# Querying VirusTotal API')
    response = requests.get(
        'https://www.virustotal.com/vtapi/v2/file/report', params=params,
headers=headers, timeout=timeout
    )

    if response.status_code == 200:
        json_response = response.json()
        vt_response_data = json_response
        return vt_response_data
    else:
        alert_output = {}
        alert_output['virustotal'] = {}
        alert_output['integration'] = 'virustotal'

        if response.status_code == 204:
            alert_output['virustotal']['error'] = response.status_code
            alert_output['virustotal']['description'] = 'Error: Public API request
rate limit reached'
            send_msg(alert_output)
            raise Exception('# Error: VirusTotal Public API request rate limit
reached')
        elif response.status_code == 403:
            alert_output['virustotal']['error'] = response.status_code
            alert_output['virustotal']['description'] = 'Error: Check credentials'
            send_msg(alert_output)
            raise Exception('# Error: VirusTotal credentials, required privileges
error')
        else:
            alert_output['virustotal']['error'] = response.status_code
            alert_output['virustotal']['description'] = 'Error: API request fail'
```

```
            send_msg(alert_output)
            raise Exception('# Error: VirusTotal credentials, required privileges
error')


def send_msg(msg: any, agent: any = None) -> None:
    if not agent or agent['id'] == '000':
        string = '1:virustotal:{0}'.format(json.dumps(msg))
    else:
        location = '[{0}] ({1}) {2}'.format(agent['id'], agent['name'], agent['ip']
if 'ip' in agent else 'any')
        location = location.replace('|', '||').replace(':', '|:')
        string = '1:{0}->virustotal:{1}'.format(location, json.dumps(msg))

    debug('# Request result from VT server: %s' % string)
    try:
        sock = socket(AF_UNIX, SOCK_DGRAM)
        sock.connect(SOCKET_ADDR)
        sock.send(string.encode())
        sock.close()
    except FileNotFoundError:
        debug('# Error: Unable to open socket connection at %s' % SOCKET_ADDR)
        sys.exit(ERR_SOCKET_OPERATION)


def get_json_alert(file_location: str) -> any:
    """Read JSON alert object from file

    Parameters
    ----------
    file_location : str
        Path to the JSON file location.

    Returns
    -------
    dict: any
        The JSON object read it.

    Raises
    ------
    FileNotFoundError
        If no JSON file is found.
    JSONDecodeError
        If no valid JSON file are used
    """
    try:
        with open(file_location) as alert_file:
            return json.load(alert_file)
    except FileNotFoundError:
        debug("# JSON file for alert %s doesn't exist" % file_location)
        sys.exit(ERR_FILE_NOT_FOUND)
    except json.decoder.JSONDecodeError as e:
        debug('Failed getting JSON alert. Error: %s' % e)
        sys.exit(ERR_INVALID_JSON)


if __name__ == '__main__':
    main(sys.argv)
```

Globalement, ce script extrait le hash généré par le FIM, et le transmet à l'API de VirusTotal en effectuant une requête HTTP POST, le hash sera ensuite comparer, grâce à l'API de VirusTotal, avec sa base de données. Cette base de données contient des informations sur des fichiers malveillant connus.

Ensuite VirusTotal renverra renverra une réponse JSON contenant l'analyse des résultats.

Appliquer la configuration

Pour appliquer la nouvelle configuration :

```
systemctl restart wazuh-manager
```

&

```
cat /var/ossec/logs/ossec/log | grep virustotal
```

Résultat :

```
root@wazu02:/var/ossec/integrations# cat /var/ossec/logs/ossec.log | grep virustotal
2025/04/03 15:31:43 wazuh-integratord: INFO: Enabling integration for: 'virustotal'.
```

VirusTotal est normalement actif.

Tests

Le fichier EICAR est un fichier de test standard utilisé pour vérifier les systèmes antivirus et de détection. Sous Windows, vous pouvez utiliser `curl.exe` pour télécharger ce fichier dans votre dossier surveillé.

```
`curl.exe -o "C:\Users\<VotreNomUtilisateur>\Documents\eicar.com"
https://secure.eicar.org/eicar.com`
```

Résultat depuis Wazuh :


```
{
  "_index": "wazuh-alerts-4.x-2025.04.04",
  "_id": "qa_gAJYBGyk3qVFy15iI",
  "_version": 1,
  "_score": null,
  "_source": {
    "input": {
      "type": "log"
    },
    "agent": {
      "ip": "@ip",
      "name": "@name",
      "id": "002"
    },
    "manager": {
      "name": "DEBwazu02"
    },
    "data": {
      "integration": "virustotal",
      "virustotal": {
        "sha1": "3395856ce81f2b7382dee72602f798b642f14140",
        "malicious": "1",
        "total": "66",
        "found": "1",
        "positives": "64",
```

```
        "source": {
          "sha1": "3395856ce81f2b7382dee72602f798b642f14140",
          "file": "c:\\users\\@user\\documents\\eicar.com",
          "alert_id": "1743771462.3656872",
          "md5": "44d88612fea8a8f36de82e1278abb02f"
        },
        "permalink":
"https://www.virustotal.com/gui/file/275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a
2c4538aabf651fd0f/detection/f-
275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f-1743770961",
        "scan_date": "2025-04-04 12:49:21"
      }
    },
    "rule": {
      "firedtimes": 1,
      "mail": true,
      "level": 12,
      "pci_dss": [
        "10.6.1",
        "11.4"
      ],
      "description": "VirusTotal: Alert - c:\\users\\@user\\documents\\eicar.com -
64 engines detected this file",
      "groups": [
        "virustotal"
      ],
      "mitre": {
        "technique": [
          "Exploitation for Client Execution"
        ],
        "id": [
          "T1203"
        ],
        "tactic": [
          "Execution"
        ]
      },
      "id": "87105",
      "gdpr": [
        "IV_35.7.d"
      ]
    },
    "location": "virustotal",
    "decoder": {
      "name": "json"
    },
    "id": "1743771464.3658204",
    "timestamp": "2025-04-04T14:57:44.671+0200"
  },
  "fields": {
    "timestamp": [
      "2025-04-04T12:57:44.671Z"
    ]
  },
  "highlight": {
    "manager.name": [
      "@opensearch-dashboards-highlighted-field@wazu02@/opensearch-dashboards-
highlighted-field@"
    ]
  },
  "sort": [
    1743771464671
  ]
}
```

VirusTotal est belle et bien implémenté.